**-- Create the database**

CREATE DATABASE university;

-- \c university;

**-- Create the student table**

CREATE TABLE student (

id SERIAL PRIMARY KEY,

name VARCHAR(100),

email VARCHAR(100) UNIQUE,

course VARCHAR(100)

);

-- Create the newStudent table

CREATE TABLE newStudent (

id SERIAL PRIMARY KEY,

name VARCHAR(100),

email VARCHAR(100) UNIQUE,

course VARCHAR(100),

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

-- Insert some values into the student table

INSERT INTO student (name, email, course) VALUES

('Alice Smith', 'alice.smith@example.com', 'Computer Science'),

('Bob Johnson', 'bob.johnson@example.com', 'Engineering'),

('Charlie Brown', 'charlie.brown@example.com', 'Business Administration');

-- 1. Trigger to insert into newStudent table after inserting into student table

CREATE OR REPLACE FUNCTION insert\_into\_newstudent()

RETURNS TRIGGER AS $$

BEGIN

INSERT INTO newStudent(name, email, course)

VALUES (NEW.name, NEW.email, NEW.course);

RETURN NEW;

END;

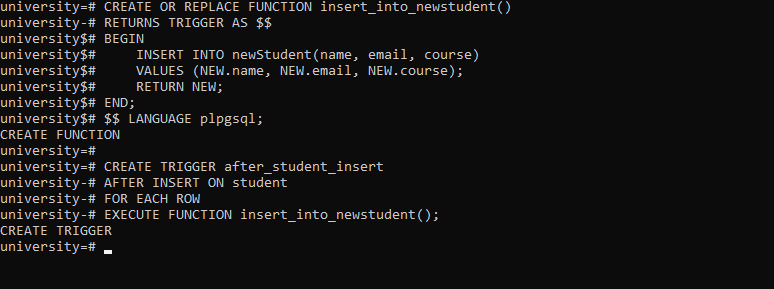
$$ LANGUAGE plpgsql;

CREATE TRIGGER after\_student\_insert

AFTER INSERT ON student

FOR EACH ROW

EXECUTE FUNCTION insert\_into\_newstudent();



-- 2. Trigger to prevent inserts on Sundays

CREATE OR REPLACE FUNCTION prevent\_insert\_on\_sunday()

RETURNS TRIGGER AS $$

BEGIN

IF EXTRACT(DOW FROM CURRENT\_DATE) = 0 THEN

RAISE EXCEPTION 'Insertions are not allowed on Sundays.';

END IF;

RETURN NEW;

END;

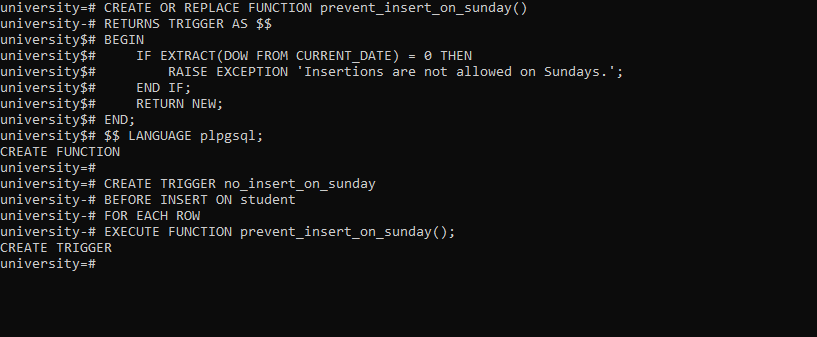
$$ LANGUAGE plpgsql;

CREATE TRIGGER no\_insert\_on\_sunday

BEFORE INSERT ON student

FOR EACH ROW

EXECUTE FUNCTION prevent\_insert\_on\_sunday();



-- 3. Trigger to UPDATE the student record in the student table.

-- This trigger will update a field in the newStudent table when a student's record is updated. Let's update the course.

CREATE OR REPLACE FUNCTION update\_student\_record()

RETURNS TRIGGER AS $$

BEGIN

UPDATE newStudent

SET course = NEW.course

WHERE email = OLD.email;

RETURN NEW;

END;

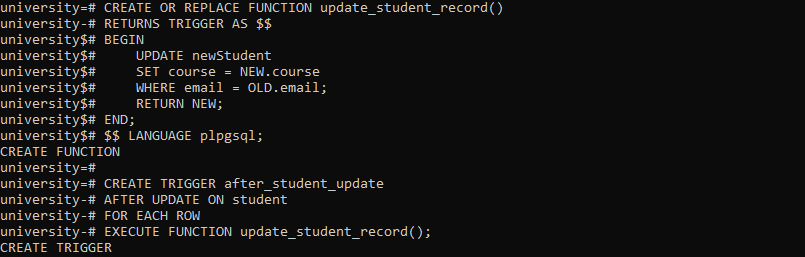
$$ LANGUAGE plpgsql;

CREATE TRIGGER after\_student\_update

AFTER UPDATE ON student

FOR EACH ROW

EXECUTE FUNCTION update\_student\_record();



-- 4. Trigger to handle deletion of a student record in the student table.

-- This trigger will insert a record into newStudent table, marking it as deleted.

CREATE OR REPLACE FUNCTION delete\_student\_record()

RETURNS TRIGGER AS $$

BEGIN

INSERT INTO newStudent (name, email, course, created\_at)

VALUES (OLD.name, OLD.email, OLD.course, CURRENT\_TIMESTAMP);

RETURN OLD;

END;

$$ LANGUAGE plpgsql;

CREATE TRIGGER before\_student\_delete

AFTER DELETE ON student

FOR EACH ROW

EXECUTE FUNCTION delete\_student\_record();

